Toward certificateless signcryption scheme without random oracles

Hu Xiong∗

∗Network and Data Security Key Laboratory of Sichuan Province, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 610054, P.R.China

Abstract

Signcryption is a useful paradigm which simultaneously offers both the functions of encryption and signature in a single logic step. It would be interesting to make signcryption certificateless to ease the heavy burden of certificate management in traditional public key cryptography (PKC) and solve the key escrow problem in Identity-based public key cryptography (ID-PKC). Most certificateless signcryption (CL-SC) schemes are constructed in the random oracle model instead of the standard model. By exploiting Bellare and Shoup’s one-time signature, Hwang et al.’s certificateless encryption and Li et al.’s identity-based signcryption, this paper proposes a new CL-SC scheme secure in the standard model. It is proven that our CL-SC scheme satisfies semantic security and unforgeability against the outside adversary and malicious-but-passive key generation center (KGC) assuming the hardness of bilinear decision Diffie-Hellman (BDDH) and computational Diffie-Hellman (CDH) problems. Our security proofs do not depend on random oracles.

Key words: Information Security; Certificateless cryptography; Signcryption; Standard model

1. Introduction

Public key cryptography (PKC) has been widely accepted and applied since it can simplify the secret key distribution problem in symmetric cryptosystem and provide a means of digital signature [16]. In traditional PKC, every user owns a public/secret key pair where the public key is usually a random string. In this case, a digital certificate issued by a trusted certification authority (CA) is needed to guarantee the relationship between the public key and the identity of the user. In general, the heavy certificate management, including certificate distribution, revocation, storage and verification, is regarded to be expensive. To remove the heavy burden of certificate management, the notion of Identity-based public key cryptography (ID-PKC) has been incepted in the cryptography community [33, 40, 21]. ID-PKC, as distinct from conventional PKC, can eliminate the need of certificates since the public key of the user can be obtained directly from its intrinsic identity information such as E-mail address or driving license number in the ID-PKC.
setting. However, ID-PKC faces the key escrow problem, i.e., a fully-trusted Private Key Generator (PKG) will generate the private key for the user according to its identity and thus can sign the document and decrypt the ciphertext on behalf of this user.

To avoid the overhead of certificate management in traditional PKC as well as the key escrow problem in ID-PKC altogether, Al-Riyami and Paterson [1] presented a new notion called certificateless public key cryptography (CL-PKC). CL-PKC can be considered as an extension of ID-PKC such that not merely the identity but also the public key will be used in the cryptographic (verification/encryption) algorithm. A Key Generation Center (KGC) is also involved to generate and distribute the private key to the user in the CL-PKC environment. Different from ID-PKC, the full secret key of each user cannot be accessed by the KGC since it is computed by the private key generated by the KGC and the secret value chosen by the user itself. In this way, the inherent key escrow problem in ID-PKC has been successfully solved in the CL-PKC environment.

Featured with confidentiality and non-repudiation, the concept of signcryption originated by Zheng [42] in 1997 has found wide applications where both confidentiality and authentication are required. Compared with the traditional signature-then-encryption approach, signcryption enjoys a lower computational cost and communication overhead. The formal definition and security proof of the signcryption has been given until 2002 by Baek et al. [4]. Furthermore, this primitive have been extensively studied in traditional PKC [5, 43, 17, 23, 41, 34, 29] and ID-PKC [28, 25, 11, 13, 14, 7, 32, 20] settings, respectively. As an extension of signcryption in the CL-PKC setting, Barbosa and Farshim [6] initialized the notion of certificateless signcryption (CL-SC) to gain the merits of CL-PKC and signcryption simultaneously. After that, several CL-SC schemes have also been proposed [2, 8, 37, 38]. The security of all these CL-SC schemes has been proved in the random oracle model [9]. In view of the criticism on the random oracle model [12], the signcryption scheme secure in the standard model receives a lot of attention. To remove the random oracles in the security proof of CL-SC, Liu et al. [27] proposed the first efficient and provably-secure CL-SC scheme in the standard model by integrating the idea of certificateless signature [26, 39] and certificateless encryption [19, 15]. However, Liu et al.’s scheme has been shown to be insecure against the outsider attack [30] and the malicious-but-passive KGC attack [36] respectively. After that, Jin et al. proposed an improvement [22] to remedy the weakness in [26]. Unfortunately, we will show that Jin et al.’s CL-SC scheme still does not offer neither semantical security against chosen ciphertext attacks nor existential unforgeability against chosen message attacks once the malicious-but-passive KGC is considered. The basic reason about our attack has also been analyzed. It is fair to say devising a CL-SC scheme secure in the standard model remains an open question until now.

In this paper, we strive to close this open problem by investigating CL-SC schemes which can be proven secure in the standard model. By exploiting Bellare and Shoup’s one-time signature [10], Hwang et al.’s certificateless encryption [19] and Li et al.’s identity-based signcryption [24], this paper proposes a new CL-SC scheme in the standard model. It is proven that our CL-SC scheme satisfies semantic security and unforgeability against outside adversary and malicious-but-passive KGC assuming the hardness of bilinear decision Diffie-Hellman (BDDH) and computational Diffie-Hellman (CDH) problems. The proofs do not rely on random oracles.

The rest of this paper is organized as follows. In Section 2, we describe the formal model of CL-SC scheme and the building blocks. In Section 3 we review and analyze Jin et al.’s CL-SC scheme. After that, our CL-SC scheme as well as the security analysis have been given in Section 4 and 5, respectively. Finally, the conclusions are given in Section 6.
2. Preliminaries

In this section, we will review the formal definition of CL-SC scheme and the building blocks of our scheme.

2.1. Definitions of CL-SC Schemes

Generally speaking, a CL-SC scheme consists of a tuple \( \text{Setup, Partial-Private-Key-Gen, User-Key-Gen, Private-Key-Gen, Signcrypt, Unsigncrypt} \) described as follows [27, 22].

1. **Setup.** Given a security parameter \( k \in \mathbb{N} \) as input, this algorithm is executed by KGC to generate the public system parameter \( \text{params} \) and a master public/secret key pair \( (mpk, msk) \).

2. **Partial-Private-Key-Gen.** Given the master secret key \( msk \) along with the user identity \( u \in \{0, 1\}^* \), this algorithm is executed by KGC to generate a user partial key \( psk_u \), which will be sent to the corresponding user securely.

3. **User-Key-Gen.** Given the public system parameter and user identity \( u \), this algorithm is executed by the user itself to generate a user public/secret key pair \( (upk_u, usk_u) \). We stress that the user secret key which will be used in the Sign algorithm cannot be accessed by the KGC to avoid the key escrow problem in ID-PKC.

4. **Private-Key-Gen.** On input \( \text{params} \), and entity’s partial private key \( psk_u \) and secret value \( usk_u \), this algorithm generates the entity’s full private key \( sk_u \). Note that this algorithm can be omitted since the full private key \( sk_u \) used in the Signcrypt or Unsigncrypt algorithms can be generated by integrating the partial private key \( psk_u \) and user secret key \( usk_u \) together in the process of performing Signcrypt or Unsigncrypt algorithms.

5. **Signcrypt.** Given the system parameters \( \text{params} \), a message \( m \), a sender’s user private key \( sk_S \), identity \( u_S \) and user public key \( upk_S \), and a receiver’s identity \( u_R \) and public key \( upk_R \), this algorithm outputs a ciphertext \( \sigma \) or an error symbol \( \bot \).

6. **Unsigncrypt.** Given a ciphertext \( \sigma \), the receiver’s user private key \( sk_R \), and the sender’s identity \( u_S \) and public key \( upk_S \), this algorithm outputs the plaintext \( m \) or an error symbol \( \bot \).

2.2. Security models

According to [6, 27], the outside attacker who can only compromise the user private key or replaces the user public key and the malicious-but-passive KGC who is responsible for the generation of the public system parameter and master public/secret key pair should be considered in the security model of CL-SC. In this way, two types of security along with two types of adversaries \( A_1 \) and \( A_2 \) has been defined for the CL-SC scheme with the restriction that \( A_1 \) cannot compromise the master secret key nor get access to the user partial key and \( A_2 \) cannot mount the key replacement attack. The oracles which can be accessed by the adversaries are described as follows.

1. **Request-Public-Key Oracle:** Given a query on identity \( u \in \{0, 1\}^* \), this oracle returns the matching user public key \( upk \).

2. **Reveal-Partial-Private-Key Oracle:** Given a query on identity \( u \), this oracle outputs the partial secret key \( psk_u \) associated with this identity.

3. **Reveal-Secret-Key Oracle:** Given a query on identity \( u \), this oracle outputs a user secret key \( usk_u \) associated with this identity.
4. **Replace-Public-Key Oracle**: Given a identity \( u \) and a new user public key \( upk_u' \), this oracle replaces the associated user’s public key with the new public key \( upk_u' \).

5. **Signcrypt Oracle**: Upon receiving a sender with identity \( u_S \), a receiver with identity \( u_R \) and a message \( m \), challenger \( C \) first runs **Signcrypt** \( (params, m, sk_S, u_S, upk_S, u_R, upk_R) \), and then returns the resulting ciphertext to the adversary. Here \( sk_S \) denotes the sender’s full private key. Note that it is possible for the challenger to be unaware of the sender’s user secret value when the associated public key has been replaced by adversary. In this case, we require the adversary to provide the sender’s user secret value.

6. **Unsigncrypt Oracle**: Upon receiving a ciphertext \( \sigma \), a sender with identity \( u_S \) and a receiver with identity \( u_R \), challenger \( C \) returns the result of **Unsigncrypt** \( (\sigma, sk_R, u_S, pk_S) \). Note that it is possible for the challenger to be unaware of the receiver’s user secret value when the associated public key has been replaced by adversary. In this case, we require the adversary to provide the receiver’s user secret value.

Regarding to the confidentiality, two games, one for \( \mathcal{A}_1 \) and the other one for \( \mathcal{A}_2 \), has been defined as follows to capture the attacks launched by adversary \( \mathcal{A}_2 \).

**Game I**: In this game, the outside attacker is modeled as Type I adversary \( \mathcal{A}_1 \) and the game simulator/challenger is modeled as \( C \).

- **Initial**. \( C \) first executes **Setup** to generate the master public/secret key pair and public system parameters, and then publishes the public system \( params \) and keeps the master secret key secret.

- **Phase 1**. In this phase, \( C \) runs \( \mathcal{A}_1 \) on \( 1^k \) and public system parameters. During the simulation, \( \mathcal{A}_1 \) can make queries onto oracles **Request-Public-Key**, **Reveal-Partial-Private-Key**, **Reveal-Secret-Key**, **Replace-Public-Key**, **Signcrypt** and **Unsigncrypt**.

- **Challenge**. Once \( \mathcal{A}_1 \) decides that **Phase 1** is over, \( \mathcal{A}_1 \) generates two equal length messages \( m_0, m_1 \), two identities \( u_{S,0} \) and \( u_{R,0} \) on which he wants to be challenged. Challenger \( C \) first chooses a bit \( \gamma \) randomly, and then computes \( \sigma^* = \text{Signcrypt}(params, m_{\gamma}, sk_{S,\gamma}, u_{S,\gamma}, upk_{S,\gamma}, u_{R,\gamma}, upk_{R,\gamma}) \). Finally, \( C \) gives \( \sigma^* \) to \( \mathcal{A}_1 \).

- **Phase 2**. Adversary \( \mathcal{A}_1 \) continues to issue queries as in **Phase 1**, and \( C \) responds in the same way as in **Phase 1**.

- **Guess**. \( \mathcal{A}_1 \) produces a bit \( \gamma' \) and wins the game if \( \gamma' = \gamma \) and the following conditions are satisfied simultaneously:
  1. \( \mathcal{A}_1 \) cannot extract the private key for any identity if the corresponding public key has been replaced.
  2. \( \mathcal{A}_1 \) cannot extract the partial private key for \( u_{R,\gamma} \) if \( \mathcal{A}_1 \) has replaced the public key \( upk_{R,\gamma} \) before the challenge phase.
  3. In **Phase 2**, \( \mathcal{A}_1 \) cannot make an unsigncryption query on the challenge ciphertext \( \sigma^* \) under \( u_{S,\gamma} \) and \( u_{R,\gamma} \) unless the sender’s public key \( upk_{S,\gamma} \) or the receiver’s public key \( upk_{R,\gamma} \), that were used to signcrypt \( m_{\gamma} \), has been replaced after the challenge phase.

The advantage of \( \mathcal{A}_1 \) is defined as \( \text{Adv}^{\text{IND-CL-SC-CCA}_2}_{\mathcal{A}_1} = |2Pr[\gamma' = \gamma] - 1| \), where \( Pr[\gamma' = \gamma] \) denotes the probability that \( \gamma' = \gamma \).

**Game II**: In this game, the insider attacker (malicious-but-passive KGC) is modeled as Type II adversary \( \mathcal{A}_2 \) and the game simulator/challenger is modeled as \( C \).
• **Initial.** $\mathcal{A}_2$ executes **Setup** to generate the master public/secret key pair and public system parameters, and send the public system $\text{params}$ and keeps the master public/secret key pair to challenger $C$. We should keep in mind that $\mathcal{A}_2$ generates $\text{params}$ and $\text{msk}$ by itself.

• **Phase 1.** During the simulation, $\mathcal{A}_2$ can make queries onto oracles **Request-Public-Key**, **Reveal-Secret-Key**, **Signcrypt** and **Unsigncrypt**. Note that $\mathcal{A}_2$ can compute the partial private key of any identity by itself with the master secret key.

• **Challenge.** Once $\mathcal{A}_2$ decides that **Phase 1** is over, $\mathcal{A}_2$ generates two equal length messages $m_0, m_1$, two identities $u_S^*$ and $u_R^*$ on which he wants to be challenged. Challenger $C$ first chooses a bit $\gamma$ randomly, and then computes $\sigma^* = \text{Signcrypt}(\text{params}, m_0, \text{sk}^*_S, u_S^*, \text{upk}^*_S, u_R^*, \text{upk}^*_R)$. Finally, $C$ gives $\sigma^*$ to $\mathcal{A}_2$.

• **Phase 2.** Adversary $\mathcal{A}_2$ continues to issue queries as in **Phase 1**, and $C$ responds in the same way as in **Phase 1**.

• **Guess.** $\mathcal{A}_2$ produces a bit $\gamma'$ and wins the game if $\gamma' = \gamma$ and the following conditions should be satisfied. In **Phase 2**, $\mathcal{A}_2$ cannot make an unsigncryption query on the challenge ciphertext $\sigma^*$ under $u_R^*$ and $u_R^*$ unless the sender’s public key $\text{upk}^*_S$, or the receiver’s public key $\text{upk}^*_R$, that were used to signcrypt $m_\gamma$, has been replaced after the challenge phase.

The advantage of $\mathcal{A}_2$ is defined as $\text{Adv}^{\text{IND-CL-SC-CCA}^2}_{\mathcal{A}_2} = |2\Pr[\gamma' = \gamma] - 1|$, where $\Pr[\gamma' = \gamma]$ denotes the probability that $\gamma' = \gamma$.

A CL-SC scheme is said to be semantically secure against adaptive chosen ciphertext attacks, if there exists neither polynomial time Type I adversary nor polynomial time Type II adversary who has a non-negligible advantage in game I and game II, respectively.

Regarding to the existential unforgeability, two games, one for $\mathcal{A}_1$ and the other one for $\mathcal{A}_2$, has also been defined as follows to capture the attacks launched by $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively.

**Game III:** Let $C$ be the game simulator/challenger with the input of security parameter $k \in \mathbb{N}$.

1. **Initial.** $C$ first executes **Setup** to generate the master public/secret key pair and public system parameters, and then publishes the public system $\text{params}$ and keeps the master secret key secret.

2. **Attack.** In this phase, $\mathcal{A}_1$ adaptively issues a polynomial bounded number of queries as in game I.

3. **Forgery.** Finally, $\mathcal{A}_1$ outputs a new triple $(\sigma^*, u_S^*, u_R^*)$, which is not produced by the Signcrypt query. Adversary $\mathcal{A}_1$ wins this game if the result of Unsigncrypt$(\sigma^*, u_S^*, \text{upk}^*_S, \text{sk}_R)$ is not the symbol $\perp$ and the queries are subject to the following constraints:
   
   (a) $\mathcal{A}_1$ cannot extract the private key for any identity if the corresponding public key has been replaced.
   
   (b) $\mathcal{A}_1$ cannot extract the partial private key for $u_S^*$ if $\mathcal{A}_1$ has replaced the public key $\text{upk}^*_R$ before the challenge phase.

The advantage of $\mathcal{A}_1$ is defined as $\text{Adv}^{\text{EUF-CL-SC-CMA}}_{\mathcal{A}_1} = \Pr[\mathcal{A}_1 \text{ wins}]$.

**Game IV:** Let $C$ be the game challenger with the input of security parameter $k \in \mathbb{N}$.

1. **Initial.** $\mathcal{A}_2$ executes **Setup** to generate the master public/secret key pair and public system parameters, and send the public system $\text{params}$ and keeps the master public/secret key pair to challenger $C$. We should keep in mind that $\mathcal{A}_2$ generates $\text{params}$ and $\text{msk}$ by itself.
2. **Attack.** In this phase, $\mathcal{A}_1$ adaptively issues a polynomial bounded number of queries as in game II.

3. **Forgery.** Finally, $\mathcal{A}_2$ outputs a new triple $(\sigma^*, u_{S^*}, u_{R^*})$, which is not produced by the Signedcrypt query. Adversary $\mathcal{A}_1$ wins this game if the result of UnSignedcrypt$(\sigma^*, u_{S^*}, up_{SK^*}, s_{sk^*})$ is not the symbol $\perp$.

The advantage of $\mathcal{A}_1$ is defined as $\text{Adv}^{\text{EUF-CL-SC-CMA}}_{\mathcal{A}_1} = \Pr[\mathcal{A}_1 \text{ wins}].$

A CL-SC scheme is said to be existentially unforgeable under adaptive chosen message attacks, if there exists neither polynomial time Type I adversary nor polynomial time Type II adversary who has a non-negligible success probability in game III and game IV, respectively.

### 2.3. Bilinear Pairing

Let $G_1$ and $G_2$ denote two multiplicative cyclic groups of prime order $p$. Let $\hat{e}$ be a bilinear map such that $\hat{e} : G_1 \times G_1 \rightarrow G_2$ with the following properties:

1. **Bilinearity:** For all $g_1, g_2 \in G$, and $a, b \in \mathbb{Z}_p$, $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.
2. **Non-degeneracy:** $\hat{e}(g_1, g_1) \neq 1_G$.
3. **Computability:** It is efficient to compute $\hat{e}(g_1, g_2)$ for all $g_1, g_2 \in G$.

**Definition 1.** Given two groups $G_1$ and $G_2$ of the same prime order $p$, a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ and a generator $g$ of $G_1$, the bilinear decision Diffie-Hellman (BDDH) problem in $(G_1, G_2, \hat{e})$ is to decide whether $Z = \hat{e}(g, g)^{abc}$ given $(g, g^a, g^b, g^c)$ and an element $Z \in G_2$. We define the advantage of a distinguisher against the BDDH problem like this

$$\text{Adv}(D) = |P_{a,b,c,\alpha} \in \mathbb{G}_p | \hat{e}(g^\alpha, g^a, g^b, g^c) = Z| - |P_{a,b,c,\alpha} \in \mathbb{G}_p | \hat{e}(g^\alpha, g^a, g^b, g^c) \neq Z|$$

**Definition 2.** Given the elements $g, g^a$ and $g^b$, for some random values $a, b \in \mathbb{Z}_p$ the Computational Diffie-Hellman (CDH) problem consists of computing the element $g^{ab}$.

### 3. Analysis of Jin et al.'s scheme

#### 3.1. Overview of Jin et al.'s scheme

Now we review Jin et al.’s [22] CL-SC scheme as follows.

**Setup.** Select a pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$ where the order of $G_1$ is $p$. Let $g$ be a generator of $G_1$. Randomly select $\alpha \leftarrow \mathbb{Z}_p, g_1 \leftarrow G_1$ and then compute $g_1 = g^\alpha$. Also select randomly the following elements: $u_i, m_i \leftarrow G_1, u_i \leftarrow G_1$ for $i = 1, \ldots, n_u, m_i \leftarrow G_1$ for $i = 1, \ldots, n_m$. Let $U = (u_1, u_2, \ldots, u_{n_u})$, $M = (m_1, m_2, \ldots, m_{n_m})$. Let $H_1 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_p$ be two collision-resistant cryptographic hash functions for some $n \in \mathbb{Z}$. The public parameters are $\text{params} = \{G_1, G_2, \hat{e}, g, g_1, g_2, U, M, H_1, H_2 \}$ and master secret is $g_2^\alpha$.

**Partial-Private-Key-Gen.** Let $u$ be a bit string of length $n_u$ representing an identity and let $u[i]$ be the $i$-th bit of $u$. Define $\mathcal{U} \subset \{1, \ldots, n_u\}$ to be the set of indices $i$ such that $u[i] = 1$. To construct the partial secret key of identity $ID$, the KGC randomly pick $r \leftarrow \mathbb{Z}_p$ and compute:

$$d_u = (g_2^\alpha (u' \prod_{i \in \mathcal{U}} u_i)^\gamma, g')$$
**User-Key-Gen.** An entity selects a secret value $x_u \leftarrow_R \mathbb{Z}_p$ as his user secret key, the public key and the corresponding signature are $(K, h, pk_u, Y, z) = (u, \hat{e}(g_1, g_2), \hat{e}(g_1, g_2)^x, y_u + cx_u \mod p)$, where $c = H_2(K, Y || \text{params})$.

**Private-Key-Gen.** The user randomly pick $r' \leftarrow_R \mathbb{Z}_p$ and compute:

$$sk_u = (sk_{u1}, sk_{u2}) = (g_{sk_5}^{x_u}(u') \prod_{i \in U} u_i)^{r'} \cdot g^g \cdot g'$$

**Signcrypt.** Verify the signature associated with the receiver’s public key by checking if the equality $h^c = Y^{pk_u}$ holds, where $c = H_2(K, Y || \text{params})$. To send a message $m \in \mathbb{G}_2$, the sender picks $r'' \leftarrow_R \mathbb{Z}_p$ and performs the following steps.

1. Compute $\sigma_1 = m \cdot \hat{e}(g_1, g_2)^{x' r''}$, $\sigma_2 = g^{r''}$, $\sigma_3 = (u' \prod_{i \in U} u_i)^{r''}$, $\sigma_4 = sk_{S,2}$.
2. Compute $m = H_1(\sigma_1, \sigma_2, \sigma_3, \sigma_4, pk_u) \in \{0, 1\}^{n_u}$ and assume $m[i]$ be the i-th bit of m and let $M \subset \{1, \ldots, n_u\}$ be the set of indices $i$ such that $m[i] = 1$.
3. Compute $\sigma_5 = sk_{S,1} \cdot (m' \prod_{j \in M} m_j)^r$.
4. Output the ciphertext $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.

**Unsigncrypt.** Given a ciphertext $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ from the user associated with an identity $u_S$ and public key $pk_S$, a verifier performs the following steps to decrypt the ciphertext:

1. Check the sender’s user public key has the right form.
2. Compute $m = H_1(\sigma_1, \sigma_2, \sigma_3, \sigma_4, pk_u) \in \{0, 1\}^{n_u}$ and assume $m[i]$ be the i-th bit of m and let $M \subset \{1, \ldots, n_u\}$ be the set of indices $i$ such that $m[i] = 1$ and check whether the following equation holds:

$$\hat{e}(\sigma_5, g) = pk_S \cdot \hat{e}(\sigma_4, u') \prod_{i \in U} u_i \hat{e}(\sigma_2, m' \prod_{j \in M} m_j).$$

If the above equation holds, output $m = \sigma_1 \cdot \hat{e}(\sigma_5, sk_{R,2})/\hat{e}(\sigma_2, sk_{R,1})$; otherwise, output $\bot$.

**Remark 1.** The algorithm **Private-Key-Gen** can be omitted since the full private key $sk_u$ generated in this algorithm can be created in **Signcrypt** and **Unsigncrypt** algorithms directly.

### 3.2. Attack against semantic security

According to [22], their scheme is semantically secure against Type I and Type II adversary in the standard model. However, we will show that their scheme is not semantically secure against chosen-ciphertext attacks by the malicious-but-passive KGC (Type II adversary $\mathcal{A}_2$) in this subsection. The attack is described in detail as follows.

1. In the initial phase, adversary $\mathcal{A}_2$ generates the public parameters $\text{params}$ and master secret key for challenger $C$. In particular, adversary $\mathcal{A}_2$ computes $m' \leftarrow_R \mathbb{G}_1$ and $m_i \leftarrow_R \mathbb{G}_1$ for $i = 1, \ldots, n_u$ as follows:
   - Choose random values $\beta^x, \beta_1, \cdots, \beta_{u_m}$ in $\mathbb{Z}_p$.
   - Compute $m' = g^g$ and $m_i = g^\beta$ for $i = 1, \ldots, n_u$.
2. In phase 1, $\mathcal{A}_2$ needs not issue any query.
3. In the challenge phase, \( \mathcal{A}_2 \) generates two equal length messages \( m_0, m_1 \), two identities \( u_S^*, u_R^* \) on which he wants to be challenged. \( \mathcal{A}_2 \) has not asked the private key extraction queries on \( u_R^* \) in Phase 1. After that, \( \mathcal{A}_2 \) sends \( m_0, m_1, u_S^*, u_R^* \) to \( C \). Then adversary \( \mathcal{A}_2 \) is given a ciphertext \( \sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*) \) such that

\[
\begin{align*}
\sigma_1^* &= m_1 \cdot e(g_1, g_2)^{x_{r'}^*}, \\
\sigma_2^* &= g^{x^*}, \sigma_3^* = (u') \prod_{i \in U_R} u_i, \sigma_4^* = sk_{S^*, 2}, \\
\sigma_5^* &= sk_{S^*, 1} \cdot (m') \prod_{j \in M} m_j,
\end{align*}
\]

where \( \gamma \) denotes the random bit chosen by the challenger \( C \), \( U_R \subset \{1, \ldots, n\} \) denotes the set of indices \( i \) such that \( u_R^*[i] = 1 \), \( m_1 = H_1(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, u_R^*, \text{pk}_R^*) \in \{0, 1\}^n \) and \( M \subset \{1, \ldots, n\} \) denotes the set of indices \( i \) such that \( m_1[i] = 1 \). Recall that the goal of \( \mathcal{A}_2 \) to win this security game is to guess \( \gamma \) correctly.

4. In phase 2, \( \mathcal{A}_2 \) randomly picks \( r' \leftarrow \mathbb{Z}_p \) and generates another ciphertext \( \sigma' = (\sigma_1', \sigma_2', \sigma_3', \sigma_4', \sigma_5') \) such that

\[
\begin{align*}
\sigma_1' &= \sigma_1^* \cdot e(g_1, g_2)^{x_r'}, \\
\sigma_2' &= \sigma_2^* \cdot g^{t'}, \sigma_3' = \sigma_3^* \cdot (u') \prod_{i \in U_R} u_i, \sigma_4' = sk_{S^*, 2}, \\
\sigma_5' &= \frac{\sigma_5^*}{(\sigma_2^*)^{y'+\sum_{j \in M} m_j}} \cdot \frac{(\sigma_2^*)^{y' + \sum_{j \in M} m_j}}{\sigma_2^*} \cdot (m') \prod_{j \in M} m_j,
\end{align*}
\]

where \( m_1' = H_1(\sigma_1', \sigma_2', \sigma_3', \sigma_4', \text{pk}_R^*) \in \{0, 1\}^n \) and \( M \subset \{1, \ldots, n\} \) denotes the set of indices \( i \) such that \( m_1'[i] = 1 \). Observe that \( \sigma' = (\sigma_1', \sigma_2', \sigma_3', \sigma_4', \sigma_5') \) is indeed a valid ciphertext under the same message \( m_1 \), the sender \( u_S^* \) and the receiver \( u_R^* \) since
In the initial phase, adversary \( A_2 \) generates the public parameters \( \text{params} \) and master secret key for challenger \( C \). In particular, adversary \( A_2 \) computes \( m' \leftarrow_R \mathbb{G}_1 \) and \( m_i \leftarrow_R \mathbb{G}_1 \) for \( i = 1, \ldots, n_m \) as follows:

- Choose random values \( \beta', \beta_1, \ldots, \beta_m \) in \( \mathbb{Z}_p \).
- Compute \( m' = g^{\beta'} \) and \( m_i = g^{\beta_i} \) for \( i = 1, \ldots, n_m \).

2. In the attack phase, \( A_2 \) issues a signcryption query by submitting a sender with identity \( u_S \), a receiver with identity \( u_R \), and a message \( m \). Then adversary \( A_2 \) is given a ciphertext
\[\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)\] such that

\[\begin{align*}
\sigma_1 &= m \cdot \hat{e}(g_1, g_2)^{h^r}, \\
\sigma_2 &= g^{r''} \cdot \sigma_3 = (u' \prod_{i \in U} u_i)^{r''}, \sigma_4 = sk_{S,2} \\
\sigma_5 &= sk_{S,1} \cdot (m' \prod_{j \in M} m_j)^{r''},
\end{align*}\]

where \(U \subset \{1, \ldots, n_u\}\) denotes the set of indices \(i\) such that \(u[i] = 1\), \(m = H_1(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_k, p_{sk_k}) \in \{0, 1\}^{n_k}\) and \(M \subset \{1, \ldots, n_m\}\) denotes the set of indices \(i\) such that \(m[i] = 1\).

From \(\sigma_2 = g^{r''}\) and \(\sigma_3 = sk_{S,1} \cdot (m' \prod_{j \in M} m_j)^{r''}\), adversary \(A_2\) can derive the sender’s full private key \(sk_{S,1}\) by computing

\[\frac{\sigma_2}{\sigma_2 \gamma_{sk_{S,1}}^{r''}} = \frac{sk_{S,1} (m' \prod_{j \in M} m_j)^{r''}}{(g^{r''} \gamma_{sk_{S,1}}^{r''})^{r''}} = \frac{sk_{S,1} (m' \prod_{j \in M} m_j)^{r''}}{(g^{r''} \gamma_{sk_{S,1}}^{r''})^{r''}} = \frac{sk_{S,1} (m' \prod_{j \in M} m_j)^{r''}}{(m' \prod_{j \in M} m_j)^{r''}}.
\]

Recall that \(\sigma_4 = sk_{S,2}\). Thus, adversary \(A_2\) can obtain the sender’s full private key \(sk_{S,1} = (sk_{S,1}, sk_{S,2})\). Equipped with sender’s full private key, \(A_2\) can definitely forge signcryption on behalf of this sender and win can always win the corresponding security game.

Our result shows that Jin et al.’s scheme can not offer semantic security and existential unforgeability against in the standard model. The basic reason of our attack is that the part of the ciphertext, where \(r\) is the blind factor in the Signcrypt algorithm.

4. Construction of our scheme

We construct a new CL-SC scheme against Type I and Type II adversaries in the standard model by incorporating the idea of Bellare and Shoup’s one-time signature [10], Hwang et al.’s certificateless encryption [19] and Li et al.’s identity-based signcryption [24]. To fight against the malicious KGC attack, our scheme use a different user public/private key generation algorithm and embed the sender’s public key in the ciphertext. In this way, the unforgeability of our scheme will be guaranteed since the secret key of the sender cannot be extracted by the malicious KGC.

**Setup.** Select a pairing \(\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2\) where the order of \(\mathbb{G}_1\) is \(p\). Let \(g\) be a generator of \(\mathbb{G}_1\). Randomly select \(a \leftarrow \mathbb{Z}_p, g_2 \leftarrow \mathbb{G}_2, K \leftarrow \{0, 1\}^j\), and then compute \(g_1 = g^a\), \(h = \hat{e}(g_1, g_2)\). Also select randomly the following elements: \(u', m' \leftarrow \mathbb{G}_1, u_i \leftarrow \mathbb{G}_1\) for \(i = 1, \ldots, n_u, m_i \leftarrow \mathbb{Z}\) for \(i = 1, \ldots, n_m\). Let \(U = \{u_i\}, M = \{m_i\}\). Let \(H_1 : \{0, 1\}^* \to \mathbb{Z}_p, H_2 : \{0, 1\}^* \to \mathbb{Z}_p, H_3 : \{0, 1\}^* \to \mathbb{Z}_p\) be three collision-resistant cryptographic hash functions for some \(n_m \in \mathbb{Z}\). The public parameters are \(params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, g_2, h, u', U, m', M, K, H_1, H_2, H_3)\) and master secret is \(g_2^a\).

**Partial-Private-Key-Gen.** Let \(u\) be a bit string of length \(n_u\) representing an identity and let \(u[i]\) be the \(i\)-th bit of \(u\). Define \(U \subset \{1, \ldots, n_u\}\) to be the set of indices \(i\) such that \(u[i] = 1\). To construct the partial secret key of identity \(ID\), the KGC randomly pick \(r_u \leftarrow \mathbb{Z}_p\) and compute:

\[(g_2^a (u' \prod_{i \in U} u_i)^{r'}, g^{r''}) = (psk_{u,1}, psk_{u,2}).\]
Therefore, the sender and the receiver’s partial private keys are

\[
(g_2^{a'} \prod_{v \in U_5} u_i)^{x_i} = (psk_{S,1}, psk_{S,2}).
\]

and

\[
(g_2^{a'} \prod_{v \in U_5} u_i)^{x_r} = (psk_{R,1}, psk_{R,2}).
\]

**User-Key-Gen.** An entity selects two secret value \(x_a, y_a \leftarrow R \mathbb{Z}_p\) as his user secret key \(usk_a\) such that \(usk_a = (x_a, y_a)\), and computes the corresponding user public key as \(upk_a = (upk_{a,1}, upk_{a,2}, upk_{a,3}) = (h^{x_a}, g_2^{y_a}, g^r)\). After that, the corresponding signature associated with the public key are computed as \(z_a = y_a + c_{1, a}x_a\) where \(c_{1, a} = H_1(K, upk_a \| params)\). According to [19], this one-time signature can be generated applying the technique of Fiat-Shamir transform without random oracles as described in [10].

**Signcrypt.** To send a message \(m \in \mathbb{G}_2\) to the receiver associated with identity \(u_R\) and user public key \(upk_R\), the sender associated with identity \(u_S\), user public key \(upk_S\), partial private key \((psk_{S,1}, psk_{S,2})\) and user secret key \(usk_S\) first checks the receiver’s user public key has the right form such that \(\hat{e}(g_1, g_2)^{x_S} = upk_{R,1}^{x_S} \cdot \hat{e}(g_1, upk_{R,2})\) and \(\hat{e}(upk_{R,2}, g) = \hat{e}(upk_{R,3}, g_2)\) where \(c_{1, R} = H_1(K, upk_R \| params)\). After that, the sender picks \(k \leftarrow R \mathbb{Z}_p\) and performs the following steps.

1. Compute \(\sigma_1 = m \cdot upk_{R,1}^{k_1}, \sigma_2 = g^k, \sigma_3 = (u')^{\prod_{v \in U_5} u_i^k}, \sigma_4 = psk_{S,2}^{x_S}\).
2. Compute \(m = H_m(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R) \in \{0, 1\}^n\) and assume \(m[i] = 0\) the \(i\)-th bit of \(m\) and let \(M \subseteq \{1, \ldots, n_m\}\) be the set of indices \(i\) such that \(m[i] = 1\).
3. Compute \(\sigma_5 = psk_{S,1}^{x_S} upk_{S,3}^{k_1} (m')^{\prod_{j \in M} m_j}\), where \(m_2 = H_2(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R)\).
4. Output the ciphertext \(\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)\).

**Unsigncrypt.** Given a ciphertext \(\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)\) from the user associated with an identity \(u_S\) and public key \(upk_S\), a verifier associated with partial private key \((psk_{R,1, psk_{R,2}})\) and user secret key \(usk_R\) performs the following steps to decrypt the ciphertext:

1. Check the sender’s user public key has the right form such that \(\hat{e}(g_1, g_2)^{x_S} = upk_{S,1}^{x_S} \cdot \hat{e}(g_1, upk_{S,2})\) and \(\hat{e}(upk_{S,2}, g) = \hat{e}(upk_{S,3}, g_2)\) where \(c_{1, S} = H_1(K, upk_S \| params)\).
2. Compute \(m = H_m(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R) \in \{0, 1\}^n\) and assume \(m[i] = 0\) the \(i\)-th bit of \(m\) and let \(M \subseteq \{1, \ldots, n_m\}\) be the set of indices \(i\) such that \(m[i] = 1\) and check whether the following equation holds:

\[
\hat{e}(\sigma_5, g) = upk_{S,1} \cdot \hat{e}(u', c_{1, S} = \hat{e}(u, c_{1, S} = \hat{e}(u, \sigma_5, upk_{S,3}^{k_1} (m')^{\prod_{j \in M} m_j}))
\]

\[
where c_2 = H_2(\sigma_1, \sigma_2, \sigma_3, \sigma_4, u_S, u_R, upk_S, upk_R).
\]

If the above equation holds, output \(m = \sigma_1 \cdot \sigma_2, psk_{R,2}^{x'_{R,2}} \sigma_3, psk_{R,1}^{x'_{R,1}}\); otherwise, output \(\bot\).

5. Analysis of our scheme

**Theorem 1.** Our CL-SC scheme is existentially unforgeable against chosen message attacks (EUF-CL-SC-CMA) in the standard model assuming the CDH problem is hard.

This theorem follows Lemmas 1 and 2.

**Lemma 1.** (Type I Existential Unforgeability). Our CL-SC scheme is $(\epsilon, \tau)$-existential unforgeable against Type I adversary with advantage at most $\epsilon$ and runs in time at most $\tau$, assuming that the $(\epsilon', \tau')$-CDH assumption holds in $\mathbb{G}_1$, where $\epsilon' \geq \frac{1}{m(na+q_i(n_a+n_m)+p+q+r+q_p)}$ and $\tau' = 1 + O(q_{pp} + q_i(n_a + n_m) + p + q_k + q_p + q_r)$ where $q_{pp}$ is the number of queries made to the Reveal-Partial-Private-Key oracle, $q_i$ is the number of queries made to the Signcrypt oracle, $q_k$ is the number of queries made to the Reveal-Secret-Key and Request-Public-Key oracles altogether, and $p$ and $\tau$ are the time for a multiplication and an exponentiation in $\mathbb{G}_1$, respectively.

**Proof.** Let $C$ be a CDH attacker who receives a random instance $(g, g^a, g^b)$ of the CDH problem in $\mathbb{G}_1$ and has to compute the value of $abP$, where $g$ is a generator of $\mathbb{G}_1$ and $a, b$ are chosen randomly from $\mathbb{Z}_p^*$. $\mathcal{A}_1$ is a type I adversary who interacts with $C$. We show how $C$ can use $\mathcal{A}_1$ as a subroutine to solve the CDH problem, i.e. to compute $abP$.

**Initial.** $\mathcal{A}_1$ sets $l_a = 2(q_i + q_k)$ and $l_m = 2q_i$, and randomly chooses two integers $k_a$ and $k_m$, with $0 \leq k_a \leq n_a$ and $0 \leq k_m \leq n_m$. We will assume that $l_a(n_a + 1) < p$ and $l_m(n_m + 1) < p$ for the given values of $q_i$, $q_k$, $n_a$, and $n_m$. The simulator then chooses an integer $x' \leftarrow \mathcal{R} \mathbb{Z}_p$ and a vector $(x_i)$ of length $n_a$, with $x_i \leftarrow \mathcal{R} \mathbb{Z}_p$ for all $i$. Likewise, it chooses another integer $z' \leftarrow \mathcal{R} \mathbb{Z}_p$ and a vector $(z_j)$ of length $n_m$, with $z_j \leftarrow \mathcal{R} \mathbb{Z}_p$ for all $j$. Lastly, $\mathcal{A}_1$ chooses two integer $y', w' \leftarrow \mathcal{R} \mathbb{Z}_p$ and two vectors, $(y_i)$ and $(w_j)$, of length $n_a$ and $n_m$, respectively, with $y_i, w_j \leftarrow \mathcal{R} \mathbb{Z}_p$ for all $i$ and $j$. Two pairs of functions are defined for an identity $u$ and a message $m$ respectively:

$$ F(u) = x' + \sum_{i \in \mathcal{U}} x_i - l_a k_a \quad J(u) = y' + \sum_{i \in \mathcal{U}} y_i \quad K(m) = z' + \sum_{j \in \mathcal{M}} z_j - l_m k_m \quad L(m) = w' + \sum_{j \in \mathcal{M}} w_j. $$

The challenger $C$ assigns $g_1 = g^{x'}, g_2 = g^{y'}, u' = g_2^{l_a k_a + x'} g^{y'}, u_i = g_2^{l_a k_a} (1 \leq i \leq n_a), m' = g_2^{l_m k_m + z'} g^{w'}, m_j = g_2^{l_m k_m} (1 \leq j \leq n_m), p = 1, \{g_k, g_2, u', (u_i), (m_j), K\}$ to $\mathcal{A}_1$. Moreover, this assignment of parameter means that the master secret will be $g_2^{l_a k_a} = g^{x'}$ and we have the following equations:

$$ u' \prod_{i \in \mathcal{U}} u_i = g_2^{F(u)} g^{J(u)} \quad \text{and} \quad m' \prod_{j \in \mathcal{M}} m_j = g_2^{K(m)} g^{L(m)}. $$

**Attack.** $C$ maintains a list $\mathcal{L} = \{u, psk_u, usk_u, upk_u, z_u\}$ which is initially empty and simulates all oracles as follows:

**Request-Public-Key Oracle:** On receiving an identity $u$ of length $n_u$, $C$ looks up the list $\mathcal{L}$ to find out the corresponding entry. If it does not exist, $C$ runs **Partial-Private-Key-Gen** and **User-Key-Gen** algorithms to generate the partial secret key $psk_u$, user public/secret key $(upk_u, usk_u)$ along with the corresponding one time signature $z_u$, respectively. Here, $upk_u = (x_u, y_u)$ and $usk_u = (upk_u,1, upk_u,2, upk_u,3) = (\hat{e}(g_2, g^{x_u}) \cdot r_u, g_1^{\frac{1}{m}} g^{x_u}).$ $z_u$ can be simulated in the same manner as in the signing oracle of the one-time signature. It then stores $(u, psk_u, usk_u, upk_u, z_u)$ into list $\mathcal{L}$. In both cases, upk_u is returned.

**Reveal-Partial-Private-Key Oracle:** On receiving a query for the partial private key of an identity $u$, $C$ can construct a partial private key by choosing $r_u \leftarrow \mathcal{R} \mathbb{Z}_p$ and computing $(psk_{u,1}, psk_{u,2}) = \left(g_1^{\frac{1}{m}} (u' \prod_{i \in \mathcal{U}} u_i)^{\frac{1}{r_u}}, g_1^{\frac{1}{m}} g^{r_u}\right)$ without knowing the master secret. Here, $F(u) \neq 0 \mod p$. It is obvious that a partial private key $(psk_{u,1}, psk_{u,2})$ associated with identity $u$ defining
in this manner is valid in case \( r_u = r_u - a/F(u) \), since that

\[
\text{psk}_{u,1} = \frac{-m_0}{g_1}(u' \prod_{i \in \mathbb{Z}} u_i)^{y_1} = g_2^{a}g_1^{F(a)}g^{F(a)}(g_2^{F(a)}g^J(a))^{y_1} = g_2^{a}(g_2^{F(a)}g^J(a))^{y_1-a/F(u)}
\]

and \( \text{psk}_{u,2} = \frac{-m_0}{g_1} \).  
\( g^a \).  
From the point of view of \( \mathbb{A}_1 \), all the partial private keys computed by \( C \) will be indistinguishable from the keys generated by a true challenger. However, \( C \) will abort the above simulation provided \( F(u) = 0 \mod p \). Assuming \( l_u(a_u + 1) < p \) which indicates \( 0 \leq l_u k_u < p \) and \( 0 \leq x + \sum_{i \in \mathbb{Z}} x_i < p \), it is trivial to know \( F(u) = 0 \mod p \) which infers that \( F(u) = 0 \mod l_u \). In this case, \( F(u) \neq 0 \mod l_u \) implies \( F(u) \neq 0 \mod p \).

Reveal-Secret-Key Oracle: On receiving a query for a public key of an identity \( u \), \( C \) looks up the list \( \mathcal{L} \) to find out the corresponding entry. If it does not exist, \( C \) runs User-Key-Gen algorithm to generate the user public/secret key pair \((upk_u, usk_u)\) and the corresponding one time signature \( z_u \) associated with the user public key \( upk_u \). It stores the key pair along with the one time signature in list \( \mathcal{L} \) and returns the secret key \( usk_u \).

Replace-Public-Key Oracle: On receiving a public key replacement request on an identity \( u \) with a new public/secret key pair \((upk'_u, usk'_u)\) and one time signature \( z'_u \), \( C \) looks up the list \( \mathcal{L} \) and updates this entry as \( \{u, psk_u, usk_u, upk_u, z_u\} \). If it does not exist, \( C \) creates a new entry for this identity by invoking the algorithm User-Key-Gen.

Signcrypt Oracle: On receiving a query for a ciphertext on a sender with identity \( u_s \), a receiver with identity \( u_R \) and a message \( m \), \( C \) first finds the items \( \{u_R, psk_R, usk_R, upk_R, z_R\} \) and \( \{u_S, psk_S, usk_S, upk_S, z_S\} \) in list \( \mathcal{L} \). After that, \( C \) checks the validity of the signature \( z_R \) of the public key \( upk_R \) and whether the user public key \( upk_S \) and the one time signature \( z_S \) have been replaced or not. If \( z_S \) is invalid or \( (upk_S, z_S) \) have been replaced, the challenger \( C \) aborts the simulation. Otherwise, \( C \) constructs a partial-secret key as the Reveal-Public-Private-Key oracle in case \( F(u_R) \neq 0 \mod l_u \). Then checks from \( \mathcal{L} \) whether the user secret key \( usk_S \) has been created or not. If it is not been created, runs the Reveal-Secret-Key oracle and stores the secret/public key pair in \( \mathcal{L} \). If it has been created, just invokes the Signcrypt algorithm to create a ciphertext on \( u_s \) and \( m \).

Unsigncrypt Oracle: On receiving a given query of an unsigncryption on a ciphertext \( \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) \), a sender associated with an identity \( u_S \), and a receiver associated with an identity \( u_R \), a public key \( upk_R = (upk_{R,1}, upk_{R,2}) \) along with the one time signature \( z_R \). \( C \) first checks the validity of the signature \( z_R \). If the verification is valid, \( C \) performs the unsigncryption as follows.

- If the public key \( upk_R \) has not been replaced, \( C \) accesses the list \( \mathcal{L} \) to find \( (psk_R, usk_R) \), performs the Unsigncrypt algorithm to recover the message \( m \), and sends it to \( \mathbb{A}_1 \). If the corresponding item does not exist, \( C \) runs the Partial-Private-Key-Gen and User-Key-Gen algorithms to generate the partial private key \( psk_R \) and the user secret key \( usk_R \) (assuming \( F(u'_R) \neq 0 \mod l_u \)), then performs the Unsigncrypt algorithm to recover the message \( m \).

- If the public key \( upk_R \) has already been replaced or \( F(u'_R) = 0 \mod l_u \), \( C \) will access the list \( \mathcal{L} \) to obtain sender \( u_S \)'s partial private key \( psk_S \) and user secret key \( usk_S = (x_S, y_S) \) such that \( upk_S = (upk_{S,1}, upk_{S,2}, upk_{S,3}) = (h^{x_S}, g_2^{y_S}, g_3) \) and retrieve receiver \( u_R \)'s user secret 


key $usk_R = (x_R, y_R)$ such that $upk_R = (upk_{R,1}, upk_{R,2}, upk_{R,3}) = (h^{x_R}, g_2^{y_R}, g_3^{x_R})$ (or the user secret key $usk_R$ associated with the current public $upk_R$ can be provided by the adversary in case the public key has been replaced). With $\sigma_3 = psk_{S,1} \cdot upk_{S,3} \cdot (K^{(m)})^{-1}$ and $\sigma_2 = g^k$ for some $k \leftarrow R \mathbb{Z}_p$ and $c_2 = H_2(\sigma_1, \sigma_2, \sigma_3, us_r, upk_k), C$ can extract $g_2^i = (\sigma_3/(psk_{S,1} \cdot \sigma_2 \cdot \sigma_3^{(m)^j}))^{1/(K^{(m)})}$ and computes $m = \sigma_1/\hat{e}(g_1, g_2^i)^{y_R}$.

**Forgery.** If C does not abort as a consequence of one of the queries above, $A_1$ will, with probability at least $\epsilon$, return the sender’s identity $us_r$, the receiver’s identity $us_r$, a message $m^*$, and valid forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*)$ on $m^*$. If $F(us_r) \neq 0 \mod p$ or $K(m^*) \neq 0 \mod p$, then C will abort. If, on the other hand, $F(us_r) = 0 \mod p$ and $K(m^*) = 0 \mod p$, C retrieves $us_r$’s user secret key $usk_r = (x_S, y_S)$ and computes

$$\frac{\sigma_2^*}{\sigma_2^*(j^m, i^*)} \cdot \frac{\sigma_2^*(j^m, i^*)}{\sigma_2^*(j^m, i^*)} = g_2^{(j^m, i^*)} \cdot \frac{g_2^{(j^m, i^*)}}{g_2^{(j^m, i^*)}} = g_2^{(j^m, i^*)},$$

where $c_2^* = H_2(\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, us_r^*, us_r^*, upk_k^*, upk_k^*)$. Thus, C can outputs $g_2^{(j^m, i^*)}$ as the solution to the CDH problem instance.

**Probability Analysis.** For the simulation to complete without aborting, we require the following conditions fulfilled:

1. All **Reveal-Partial-Private-Key** queries on an identity $u$ have $F(u) \neq 0 \mod l_u$.
2. All **Signencrypt** queries of an sender $us_r$ have $F(us_r) \neq 0 \mod l_u$.
3. All **Unsigncrypt** queries of a receiver $us_r$ will either have $F(us_r) \neq 0 \mod l_u$ or $K(m) \neq 0 \mod l_m$ where $m = H_m(\sigma_1, \sigma_2, \sigma_3, \sigma_4, us_r, upk_k) \in \mathbb{Z}$.
4. $F(u_r) = 0 \mod l_u$ and $K(m^*) = 0 \mod l_m$.

In order to make the analysis more simple, we will bound the probability of a subcase of this event. Let $u_1, \ldots, u_{q_r}$ be the identities appearing in these queries not involving the challenge identity and let $m_1, \ldots, m_{q_M}$ be the messages in the unsigncript queries involving the challenge identity $u^*$. Clearly, we will have $q_1 \leq d_{pp} + q_s + q_u$ and $q_M \leq q_u$. Define the events $A_i, A^*, B_j, B^*$ as

$$A_i : F(u_i) \neq 0 \mod l_u, \quad i = 1, \ldots, q_t, \quad A^* : F(u^*_r) = 0 \mod p$$

$$B_j : K(m_j) \neq 0 \mod l_m, \quad j = 1, \ldots, q_M, \quad B^* : K(m^*) = 0 \mod l_m$$

The probability of C not aborting is $\Pr[\text{not abort}] \geq \Pr[\bigwedge_{i=1}^{q_t} A_i \land A^* \land \bigwedge_{j=1}^{q_M} B_j \land B^*]$. It is obvious to observe that the events $\bigwedge_{i=1}^{q_t} A_i \land A^*$ and $\bigwedge_{j=1}^{q_M} B_j \land B^*$ are independent.

The assumption $l_u(n_u + 1) < p$ implies if $F(u) = 0 \mod p$ then $F(u) = 0 \mod l_u$. In addition, it also implies that if $F(u) = 0 \mod l_u$, there will be a unique choice of $k_u$ with $0 \leq k_u \leq n_u$ such that $F(u) = 0 \mod p$. Since $k_u, x^*$ and vector $(x)$ of length $n_u$ are randomly chosen, we have

$$Pr[A^*] = Pr[F(u^*) = 0 \mod p \land F(u^*) = 0 \mod l_u]$$

$$= Pr[F(u^*) = 0 \mod l_u]Pr[F(u^*) = 0 \mod p] Pr[F(u^*) = 0 \mod l_u] = \frac{1}{l_u} \frac{1}{n_u + 1}$$
On the other hand, we have \( \Pr[\bigwedge_{i=1}^{q_i} A_i | A^*] = 1 - \Pr[\bigvee_{i=1}^{q_i} \lnot A_i | A^*] \geq 1 - \sum_{i=1}^{q_i} \Pr[\lnot A_i | A^*] \) where \( \lnot A_i \) denote the event \( F(u_i) = 0 \mod l_u \).

If \( F \) is evaluated on two different identities, \( u_{i1} \) and \( u_{i2} \), then the sums appearing in \( F(u_{i1}) \) and \( F(u_{i2}) \) will differ in at least one randomly chosen value, and the events \( F(u_{i1}) = 0 \mod l_u \) and \( F(u_{i2}) = 0 \mod l_u \) will be independent. Also since the events \( A_i \) and \( A^* \) are independent for any \( i \), we have \( \Pr[\lnot A_i | A^*] = 1/l_u \). Hence, we have

\[
\Pr[\bigwedge_{i=1}^{q_i} A_i \land A^*] = \Pr[A^*] \Pr[\bigwedge_{i=1}^{q_i} A_i | A^*] \geq \frac{1}{l_u(n_u + 1)}(1 - \frac{q_{pp} + q_s + q_u}{l_u})
\]

and setting \( l_u = 2(q_{pp} + q_s + q_u) \) as in the simulation gives \( \Pr[\bigwedge_{i=1}^{q_i} A_i \land A^*] \geq \frac{1}{4(q_{pp} + q_s + q_u)n_u + 1} \).

A similar analysis for the sign queries gives the result \( \Pr[\lnot A_i | A^*] \geq \frac{1}{4(q_{pp} + q_s + q_u)n_u + 1} \).

If the simulation does not abort, \( \mathcal{A}_1 \) will produce a forged signature with probability at least \( \epsilon \). Thus \( C \) can solve for the CDH problem instance with probability \( \epsilon' = \frac{1}{4(q_{pp} + q_s + q_u)n_u + 1} \).

**Lemma 2.** (Type II Existential Unforgeability). Our CL-SC scheme is \( (\epsilon, t) \)-existential unforgeable against Type II adversary with advantage at most \( \epsilon \) and runs in time at most \( t \), assuming that the \( (\epsilon', t') \)-CDH assumption holds in \( G_1 \), where \( \epsilon' = t + O((q_{pp} + q_s)n_u + q_u)/p + (q_s + q_u)t') \).

**Proof.** Let \( C \) be a CDH attacker who receives a random instance \( (g, g^a, g^b) \) of the CDH problem in \( G_1 \) and has to compute the value of \( ab \), where \( g \) is a generator of \( G \); \( a, b \) are chosen randomly from \( \mathbb{Z}_q^* \). \( \mathcal{A}_2 \) is a type II adversary who interacts with \( C \). We show how \( C \) can use \( \mathcal{A}_2 \) as a subroutine to solve the CDH problem, i.e. to compute \( ab \).

**Initial.** \( C \) sets \( l_u = 2(q_{pp} + q_s) \) and \( l_m = 2q_s \), and randomly chooses two integers \( k_u \) and \( k_m \), with \( 0 \leq k_u \leq n_u \) and \( 0 \leq k_m \leq n_m \). We will assume that \( l_u(n_u + 1) < p \) and \( l_m(n_m + 1) < p \) for the given values of \( q_{pp} \), \( q_s \), \( n_u \) and \( n_m \). The simulator then chooses an integer \( x' \leftarrow_R \mathbb{Z}_q \\n \) and a vector \( (x_i) \) of length \( n_u \), with \( x_i \leftarrow_R \mathbb{Z}_q \\n \) for all \( i \). Likewise, it chooses another integer \( x' \leftarrow_R \mathbb{Z}_q \\n \) and a vector \( (x_i) \) of length \( n_m \), with \( x_i \leftarrow_R \mathbb{Z}_q \\n \) for all \( j \). Lastly, \( C \) chooses two integer \( y', w' \leftarrow_R \mathbb{Z}_q \\n \) and two vectors, \( (y_i) \) and \( (w_j) \), of length \( n_u \) and \( n_m \), respectively, with \( y_i, w_j \leftarrow_R \mathbb{Z}_q \\n \) for all \( i \) and \( j \). Two pairs of functions are defined for an identity \( u \) and a message \( m \) respectively:

\[
F(u) = x' + \sum_{i \in \mathcal{U}} x_i - l_u k_u, \quad J(u) = y' + \sum_{i \in \mathcal{U}} y_i, \quad K(m) = z' + \sum_{j \in \mathcal{M}} z_j - l_m k_m, \quad L(m) = w' + \sum_{j \in \mathcal{M}} w_j.
\]

The challenger \( C \) selects \( \alpha \leftarrow_R \mathbb{Z}_q \\n \) and assigns \( g_1 = g^\alpha \), \( g_2 = g^b \), \( u' = g_2^{-(k_u + x')} \), \( g^{\alpha'} \), \( m' = g_2^{-(k_m + z')} \), \( g^{\alpha'} \), \( m_j = g_2^{-(k_m + z')} \), \( 1 \leq j \leq n_m \), \( K \leftarrow_R \mathbb{Z}_q \) \( (0, 1)^2 \), and sends the system parameters \( \text{params} = (g_1, g_2, u', (u_i), (m'), (m_j), K) \) as well as the master secret \( g_2^a = (g^b)^\alpha \) to \( \mathcal{A}_2 \). Moreover, this assignment of parameter means that:

\[
U = u' \prod_{i \in \mathcal{U}} u_i = g_2^{F(u)} g^{\alpha u} \quad \text{and} \quad m' \prod_{i \in \mathcal{M}} m_i = g_2^{K(m)} g^{\alpha m}.
\]

**Attack.** \( C \) maintains a list \( \mathcal{L} = \{u, psk_u, usk_u, upk_u, z_u\} \) which is initially empty and simulates all oracles as follows:

**Request-Public-Key-Oracle:** On receiving an identity \( u \) of length \( n_u \), \( C \) looks up the list \( \mathcal{L} \) to find out the corresponding entry. If it does not exist, \( C \) runs **Partial-Private-Key-Gen** and **User-Key-Gen** algorithms to generate the partial secret key \( psk_u \), user public/secret key \( (upk_u, usk_u) \).
If the public key \( u \in \mathbb{Z}_p \) can construct a partial private key by choosing \( r_u \leftarrow \mathbb{Z}_p \) randomly and computing \((psk_{a1}, psk_{a2}) = ((g^\alpha)^{-1}\prod_{i \in U} u_i)^{r_u}, \left((g^\alpha)^{-1}\prod_{i \in U} u_i\right)^{\alpha} = (g^\alpha)^{r_u}(u'_i \prod_{i \in U} u_i)^{\alpha}, (g^\alpha). \) Here, \( F(u) \neq 0 \) mod \( p \). It is obvious that a partial private key \((psk_{a1}, psk_{a2})\) associated with identity \( u \) defining in this manner is valid in case \( R_u = r_u - \alpha e/F(u) \). From the point of view of \( \mathcal{A}_2 \), all the partial private keys computed by \( C \) will be indistinguishable from the real one. However, \( C \) will abort the above simulation provided \( F(u) = 0 \) mod \( p \). Assuming \( l_u(n_u + 1) < p \) which indicates \( 0 \leq l_u < p \) and \( 0 \leq x_i + \sum_{i \in U} x_i < p \), it is trivial to know \( F(u) = 0 \) mod \( p \) which infers that \( F(u) = 0 \) mod \( l_u \). In this case, \( F(u) \neq 0 \) mod \( l_u \) implies \( F(u) \neq 0 \) mod \( p \).

Reveal-Secret-Key Oracle: On receiving a query for a public key of an identity \( u \in U \), \( C \) looks up the list \( L \) to find out the corresponding entry. If it does not exist, \( C \) runs User-Key-Gen algorithm to generate the user public/secret key pair \((upk_u,usk_u)\) and the corresponding one time signature \( z_u \) associated with the user public key \( upk_u \). It stores the key pair along with the one time signature in list \( L \) and returns the secret key \( usk_u \).

Signcrypt Oracle: On receiving a query for a ciphertext on a sender with identity \( u_5 \), a receiver with identity \( u_R \) and a message \( m \), \( C \) first finds the items \([upk_u, psk_u, usk_u, upk_R, z_R]\) and \([us, psks, usks, upks, zs]\) in list \( L \). After that, \( C \) checks the validity of the one time signature \( z_R \) of the public key \( upk_R \) and whether the user public key \( upk_u \) and the one time signature \( z_u \) have been replaced or not. If \( z_u \) is invalid or \((upks, zs)\) have been replaced, the challenger \( C \) aborts the simulation. Otherwise, \( C \) constructs a partial-secret key as in the Reveal-Partial-Private-Key oracle in case \( F(u(s)) \neq 0 \) mod \( l_u \). \( C \) then checks from \( L \) whether the user secret key \( usk_u \) has been created or not. If it is not been created, runs the Reveal-Secret-Key oracle and stores the secret/public key pair in \( L \). If it has been created, it just invokes the Signcrypt algorithm to create a ciphertext on \( u \) and \( m \).

Unsigncrypt Oracle: On receiving a given query of an unsigncryption on a ciphertext \( \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) \), a sender associated with an identity \( u_5 \), and a receiver associated with an identity \( u_R \), a public key \( upk_R = (upk_{R1}, upk_{R2}) \) along with the one time signature \( z_R \), \( C \) first checks the validity of the signature \( z_R \). If the verification is valid, \( C \) performs the unsigncryption as follows.

- If the public key \( upk_R \) has not been replaced, \( C \) accesses the list \( L \) to find \((psk_R, usk_R)\), performs the Unsigncrypt algorithm to recover the message \( m \), and sends it to \( \mathcal{A}_1 \). If the corresponding item does not exist, \( C \) runs the Partial-Private-Key-Gen and User-Key-Gen algorithms to generate the partial private key \( psk_R \) and the user secret key \( usk_R \) (assuming \( F(u'_R) \neq 0 \) mod \( l_u \)), then performs the Unsigncrypt algorithm to recover the message \( m \).

- If the public key \( upk_R \) has already been replaced or \( F(u'_R) = 0 \) mod \( l_u \), \( C \) will access the list \( L \) to obtain sender \( u_5 \)'s partial private key \( psk_5 \) and user secret key \( usk_5 = (x_5, y_5) \) such that \( upk_5 = (upk_{S1}, upk_{S2}, upk_{S3}) = (h^{x_5}, g^{x_5^y}, g^{x_5^y}) \) and retrieve receiver \( u_R \)'s user secret key \( usk_R = (x_R, y_R) \) such that \( upk_R = (upk_{R1}, upk_{R2}, upk_{R3}) = (h^{x_5}, g^{x_5^y}, g^{x_5^y}) \) (or the user secret key \( usk_R \) associated with the current public \( upk_R \) can be provided by the adversary in case the public key has been replaced). With \( \sigma_5 = psk_{S5} \cdot upk_{S3} \cdot (g^{x_5^y} \cdot g^{x_5^y}) \) and
\[ \sigma_2 = g^k \text{ for some } k \leftarrow_R \mathbb{Z}_p \text{ and } c_2 = H_2(\sigma_1, \sigma_2, \sigma_4, u_S, u_R, upk_S, upk_R), C \text{ can extract} \]
\[ g_2^{c_2} = (\sigma_5^{n_1 \cdot \sigma_2^{n_2} \cdot \sigma_4^{n_4} \cdot \sigma_5^{n_5})^{1/(K(m))} \text{ and computes } m = \sigma_1 / (g_1, g_2^{c_2}). \]

**Forgery.** If C does not abort as a consequence of one of the queries above, \( \mathcal{A}_2 \) will, with probability at least \( \epsilon \), return the sender’s identity \( u_S \), the receiver’s identity \( u_R \), a message \( m^* \), and valid forgery \( \sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_4^*, \sigma_5^*) \) on \( m^* \). If \( F(u_S) \neq 0 \mod p \) or \( K(m^*) \neq 0 \mod p \), then C will abort. If, on the other hand, \( F(u_S) = 0 \mod p \) and \( K(m^*) = 0 \mod p \), C retrieves \( u_S \)'s user secret key \( usk_S = (x_S, y_S) \) and computes
\[ \frac{\sigma_5^*}{\sigma_2^* \cdot \sigma_4^* \cdot c_2} = \frac{g_2^{auxx^*} \cdot (u' \prod_{i \in U, u_i} b_i \cdot (m' \prod_{j \in M} m_j)^{ab} \cdot g_{xy^*}^{c_2}}{g_{uxy_1} \cdot g_{L(m) u_c} \cdot g_{yx^*_1} \cdot c_2} = g_{auxx^*}^{c_2}, \]
where \( c_2^* = H_2(\sigma_1^*, \sigma_2^*, \sigma_4^*, \sigma_5^*, u_S, u_R, upk_S, upk_R) \). Thus, C can outputs \( g_2^{ab} \) as the solution to the CDH problem instance.

The probability analysis is similar to the proof in Lemma 1 to avoid repetition.

**Theorem 2.** Our CL-SC scheme is indistinguishable against chosen ciphertext attacks (IND-CL-SC-CCA) in the standard model assuming the decisional BDDH problem is hard.

This theorem follows Lemmas 3 and 4.

**Lemma 3.** (Type I confidentiality). Our CL-SC scheme is \( (\epsilon, t) \)-indistinguishable against Type I chosen ciphertext adversary with advantage at most \( \epsilon \) and runs in time at most \( t \), assuming that the \( \epsilon' \)-, \( t' \)-BDDH assumption holds in \( \mathcal{G}_1 \), where \( \epsilon' = \frac{\epsilon}{32(q_e n_u + q_i n_m + q_i n_u + q_i n_m + 1)} \) and \( t' = t + O((q_e n_u + q_i n_m) \cdot \log(q_e n_u + q_i n_m) \cdot \log(q_e n_u + q_i n_m) \cdot \log(q_e n_u + q_i n_m)) \).

**Proof.** Let C be a BDD attacker who receives a random instance \( (g, g^a, g^b, g^c, Z) \) of the BDH problem and has to output a guess \( \beta \), to show whether the challenge is a BDH tuple. Here \( g \) is a generator of \( \mathcal{G}_1 \), \( Z \) is randomly chosen from \( \mathcal{G}_1 \) and \( a, b, c \) are chosen randomly from \( \mathbb{Z}_p \). \( \mathcal{A}_1 \) is a type I adversary who interacts with C. We show how C can use \( \mathcal{A}_1 \) as a subroutine to solve the BDH problem.

**Initial.** C first sets the system parameters described in Lemma 1. Note that in the **Initial** phase, C assigns \( g_1 = g^a \), \( g_2 = g^b \). After that, C defines the functions \( F(u), J(u), K(m), L(m) \) and \( u', (u_i), m', (m_j) \) such that
\[ u' \prod_{i \in U} u_i = g_2^{F(u)} \text{ and } m' \prod_{j \in M} m_j = g_2^{K(m)} \cdot g^{L(m)} \]

**Phase 1.** \( \mathcal{A}_1 \) can perform a polynomially bounded number of queries including the Request-Public-Key, Reveal-Partial-Private-Key, Reveal-Secret-Key, Replace-Public-Key, Signcrypt and Unsigncrypt queries. The challenger C answers the queries of \( \mathcal{A}_1 \) identical to Lemma 1.

**Challenge.** \( \mathcal{A}_1 \) generates two equal length messages \( m_0, m_1 \), two identities \( u_S^* \) and \( u_R^* \) on which he wants to be challenged. \( \mathcal{A}_1 \) has not asked the private key extraction queries on \( u_R^* \).

**In Phase 1.** After that, \( \mathcal{A}_1 \) sends \( m_0, m_1 \) and \( u_S^*, u_R^* \) to C. If \( F(u_R^*) \neq 0 \mod u_c \), C aborts the simulation. Otherwise, C takes a bit \( y \in \{0, 1\} \) and constructs a ciphertext of \( m_y \) as follows. Let
\[ upk_S = (upk_{S,1}, upk_{S,2}, upk_{S,3}) = (h^{x_S}, g_2^{x_S}, g^{x_S}) \text{ and } upk_R = (upk_{R,1}, upk_{R,2}, upk_{R,3}) = (h^{x_R}, g_2^{x_R}, g^{x_R}) \text{ be } u_S^* \text{ and } u_R^* \text{ current user public keys. } C \text{ retrieves the corresponding user secret keys } usk_S^* = (x_S, y_S^*) \text{ and } usk_R^* = (x_R, y_R^*). \]
Then C sets \( \sigma_1^* = m_y \cdot Z^e, \sigma_2^* = g^e, \sigma_3^* = \]
Lemma 4. (Type II confidentiality). Our CL-SC scheme is $(\epsilon, \eta)$-indistinguishable against Type II chosen ciphertext adversary with advantage at most $\epsilon$ and runs in time at most $t$, assuming that the $(\epsilon', t')$-BDH assumption holds in $\mathcal{G}_1$, where $\epsilon' \geq \frac{2m_0 + m_1}{\log q} + q(t_0 + q_0 + n_0 + n_0 + 1) + n_0 + 1$ and $t' = t + O(t_0 n_0 + q_0 (n_0 + n_0) + (q_0 + q_1) \eta)$.

Proof. Let $C$ be a BDH attacker who receives a random instance $(g, g^a, g^b, g^c, Z)$ of the BDH problem and has to output a guess $\hat{\beta}$, to show whether the challenge is a BDH tuple. Here $g$ is a generator of $\mathcal{G}_1$, $Z$ is randomly chosen from $\mathcal{G}_1$ and $a, b, c$ are chosen randomly from $\mathbb{Z}_p^*$. $\mathcal{A}_2$ is a Type II adversary who interacts with $C$. We show how $C$ can use $\mathcal{A}_2$ as a subroutine to solve the BDH problem.

Initial. $C$ first sets the system parameters described in Lemma 2. Note that in the Initial phase, $C$ selects $\alpha \leftarrow \mathbb{Z}_p$ and assigns $g_1 = g^a$, $g_2 = g^b$. After that, $C$ defines the functions $F(u), J(u), K(m), L(m)$ and $u', (u_i), (m_i), (m_i)$ such that

$$u' \prod_{i \in \mathcal{U}} u_i = g_2^{F(u)} g_1^{J(u)} \quad \text{and} \quad m' \prod_{i \in M} m_i = g_2^{K(m)} g_1^{L(m)}$$

Phase 1. $\mathcal{A}_2$ can perform a polynomially bounded number of queries including the Request-Public-Key, Reveal-Partial-Private-Key, Reveal-Secret-Key, Signcrypt and Unsigncrypt queries. The challenger $C$ answers the queries of $\mathcal{A}_1$ identical to Lemma 2.

Challenge. $\mathcal{A}_2$ generates two equal length messages $m_0, m_1$, two identities $u_{s^*}$ and $u_{r^*}$ on which it wants to be challenged. $\mathcal{A}_2$ has not asked the private key extraction queries on $u_{r^*}$ in Phase 1. After that, $\mathcal{A}_2$ sends $m_0, m_1$ and $u_{s^*}, u_{r^*}$ to $C$. If $F(u_{r^*}) \neq 0 \mod \ell$, $C$ aborts the simulation. Otherwise, $C$ takes a bit $\gamma \leftarrow \{0, 1\}$ and constructs a ciphertext of $m_y$ as follows. Let $upk_{s^*} = (upk_{s^*}, 1, upk_{s^*}, 2, upk_{s^*}, 3) = (\hat{\beta}(g_2, g^a r_{s^*}, g_2^b, g^c) \quad \text{and} \quad upk_{r^*} = (upk_{r^*}, 1, upk_{r^*}, 2, upk_{r^*}, 3) = (\hat{\beta}(g_2, g^a r_{r^*}, g_2^b, g^c))$ be $u_{s^*}$ and $u_{r^*}$’s current user public keys. $C$ retrieves the corresponding user secret keys $usk_{s^*} = (x_{s^*}, y_{s^*})$ and $usk_{r^*} = (x_{r^*}, y_{r^*})$. Then $C$ sets $\sigma_{s^*} = m_y \cdot Z_{s^*}^{-x_{s^*}}$, $\sigma_{r^*} = g^\gamma$, $\sigma_{s^*} = (g^{\gamma^{\lambda_{s^*}}})^{-x_{s^*}}$ and computes $\sigma_{s^*} = (g^{\gamma^{\lambda_{s^*}}})^{-x_{s^*} / F(usk_{s^*})}$. Let $m_y = H_m(\sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, u_{s^*}, u_{r^*}, upk_{s^*}, upk_{r^*}) \in \{0, 1\}^{m_0}$ and assume $m_y[i]$ be the $i$-th bit of $m_y$ and let $M_y \subset \{1, \ldots, m_y\}$ be the set of indices $i$ such that $m_y[i] = 1$. If $K(m_y) \neq 0 \mod \ell$, $C$ aborts the simulation. Otherwise, $C$ sets $\sigma_{s^*} = ((g^{\gamma^{\lambda_{s^*}}})^{-x_{s^*} / F(usk_{s^*})})^{\lambda_{s^*}}(u' \prod_{i \in \mathcal{U}} u_i)^{\gamma_{s^*} (g^{\gamma^{\lambda_{s^*}}})^{\lambda_{s^*}} \prod_{i \in M} m_i}$ and returns $\sigma^* = (\sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*})$ to $\mathcal{A}_1$. Here, $c^*_i = H_2(\sigma_{s^*}, \sigma_{r^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, \sigma_{s^*}, u_{s^*}, u_{r^*}, upk_{s^*}, upk_{r^*})$.
Guess. Finally, $A_1$ produces a bit $\gamma'$. If $\gamma' = \gamma$, $C$ wins the game and indicates that $Z = \hat{e}(g, g)^{\alpha c}$. Otherwise, $C$ fails to solve the BDH problem.

The probability analysis is similar to the proof in Lemma 1 to avoid repetition.

6. Conclusion

It is desirable to devise CL-SC schemes secure in the standard model. We have showed that Jin et al.’s CL-SC scheme is not secure against the malicious KGC attacks. In this paper, motivated by Bellare and Shoup’s one-time signature, Hwang et al.’s certificateless encryption and Li et al.’s identity-based signcryption, we proposed an CL-SC scheme provably secure in the standard model. It is believed to be the first in the literature to achieve the provably secure CL-SC without random oracles. Our future work consists of constructing CL-SC scheme secure in the standard model without one-time signature.

Acknowledgement

This work is partially supported by National Natural Science Foundation of China under Grant Nos. 61003230, 6137026, 61300191 and 61272029, Chongqing Key Lab of Computer Network and Communication Technology under Grant No. CY-CNCL-2012-02, National Key Technology R&D Program (2009BAG12A10), China Railway Ministry major task (2008G017-A) and the State Key Laboratory of Rail Traffic Control and Safety (RCS2009ZT007). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References


H. Xiong, Z. Qin, F. Li, New Identity-based Three-party authenticated key agreement protocol with Provable
